Maria-2 handy hints 10/03/83

This little document should help out in converting your programs to work on Maria-2. The main difference between Marias one and two is that a kernel is no longer necessary. Instead, our graphics chip figures out what it is supposed to be displaying by looking at something called, for lack of a better term, the display list list (henceforth referred to as DLL). This is, as it sounds, a list telling Maria where our display lists are in memory (I know this is elementary to many of you but I wanted to cover all bases). In addition this DLL informs Maria of the height of this zone, whether or not to trigger a display list interrupt (old hat to you 5200 programmers) and controls "HOLY DMA." Detailed discussions of these concepts are available in the Tom Westberg Papers (Random House paperback).

Excerpted from these papers is the following information

DLL format is as follows:

<dli> <a12en,a11en,X> <offset -- 4 bits>

<dph> -- display list pointer high

<dpl> -- display list pointer low

Note that registers dpl and dph are no longer needed externally. To avoid confusion the pointers to the DLL resides at there previous addresses. If the dli bit is set, a non-maskable interrupt (NMI) will occur. It is up to the programmer to determine which interrupt just occurred. A12en should be set if you wish to utilize holy dma for sixteen raster zones. When accessing graphics rom while this is set, Maria will "see" zeros at addresses that have both A15 and A12 set. A11en should be set if you want holy dma for eight line zones. Please note that you really can't use this feature yet, as current ram carts go from $4000 -- $8000 or $A000. Because maria also wants A15 to be high for holy dma, graphics must be located above $8000 for eight high zones, and above $A800 for sixteen high zones. The moral of this story is, don't kiss your zeros goodbye -- yet. New ram cards are in the works. The Offset value is the Zonelength-1. In other words put $F here for sixteen high, $7 for eight high, etc. The X is don't care, but should probably be zero.

For Maria to know where the Heck DLL is, you must store its start address in the registers DPPL and DPPH. As I mentioned earlier, these reside in the same place the late great DPL and DPH lived. This only has to be done once at initialization. IT IS VERY IMPORTANT THAT THESE REGISTERS BE STORED BEFORE DMA IS STARTED BY STORING TO CTRL. Otherwise things could get messy as maria tries to dma out of space.

By the way, our old friend CTRL is a bit different now. Here's the scoop:

CK DM1 DM0 CWIDTH BCNTL KM RM1 RM0

Where:

CK: Color Kill. Makes things go B/W on you. A one here kills color

DMn. Dma mode. Explained below.

CWIDTH: A zero means one byte characters, a one means two.

BCNTL: Border control. One means bleed the background color to the edge

KM: Kangaroo mode. A one turns off transparency. Useful for 320x2 mode.

RM: Read mode. Inverted from Maria one.

DM1	DM0	Meaning	RM1	RM0	Meaning
0	0	Test A	1	1	320A
0	1	Test B	1	0	320B
1	0	Run normally	0	1	Reserved
1	1	Inactive	0	0	160x2 or 160x4

The test modes should never be invoked, as your 6502 will get very confused when its address lines start going all over the place without the HALT line asserted during DMA. Only use the 1 0 or 1 1.

I reiterate -- the initial value of CTRL must be stored AFTER DPPH and DPPL have been set up. Additionally, you should do all of your other initialization first, wait for vblank to start, and then store $40 or whatever to CTRL.

Display list headers have changed once again. There are two forms that they may assume -- the short (1040 EZ) and the long (1040).

Shorty:
```
< ppl >    -- low address of graphics referenced
< w  >    -- width = <P P P> <W W W W W>
< pph >    -- high pointer to graphics
< hpos>    -- you guessed it -- horizontal position
```

The short header is used only for direct mode graphics. Note also that we now have a full five bits of width (this applies also to character maps). You might ask, "whatever happened to the indirect bit?" Well Virginia, here comes the five byte header:

```
< ppl >
< w1 >     -- <wm> <1> <ind> <0 0 0 0 0>
< pph >
< w  >     -- <P P P> <W W W W W>
< hpos>
```

wm = the new value of the write mode from now on. 0 ⇒ 160A×2, 320×1 (4 bytes = 4 pixel cells)

ind = indirect mode for this header only.

As usual, DMA is ended by a header with a second byte of zero. An amusing side effect of this with the five byte header is that a value of zero in the < w > byte will give you a thirty-two byte wide object. The long header is used to change the write mode, for thirty-two wide objects, and for all character map headers.

Maria starts DMAing out of your DLL on the (I believe) 16th line after vsync. This is a quite a few rasters before the visible screen known and loved by all. Additionally, Maria stops reading on the 256th raster -- some time after our regulation (192) sized screen ends. This leaves 50 rasters to be split between the top and bottom of the screen. DLL elements must be in place for this space, probably pointing to a couple of zeros so not much DMA will happen. Note that these DLL spots are a good place to put

a couple of DLIs — for vblank and top of screen type stuff. Some standard should be set for how many lines go above and below the visible screen. For now, I'm splitting the leftovers evenly with 25 above and below.

Other than these details, much remains the same. The only other changes are:

You no longer have to convert the graphics to wierd bit patterns

The memory map has changed somewhat. See the attached chart for details.

Disco boogie git down! Have fun programming the most awesome game machine in history!

-kevino

3600 MODE MEMORY MAP

(1702 version)